ERDC MSRC Major Shared Resource Center

ERDC MSRC PET Technical Report No. 01-06

Parallel Finite Element Simulation of Wave Interacting with Ships in Motion

by

Shahrouz Aliabadi
Andrew Johnson
Bruce Zellars
Charlie Berger
Jane Smith

20 April 2001

Work funded by the Department of Defense
High Performance Computing Modernization Program
U.S. Army Engineer Research and Development Center
Major Shared Resource Center through

Programming Environment and Training

Supported by Contract Number: DAHC94-96-C0002
Computer Sciences Corporation

Parallel Finite Element Simulation of Waves Interacting with Ships in Motion

Shahrouz Aliabadi[1], Andrew Johnson[2], Bruce Zellars[1], Charlie Berger[3] and Jane Smith[3]

1 – Department of Engineering, Clark Atlanta University
223 James P. Brawley Dr. S. W., Atlanta, Georgia 30314
Email: aliabadi@cau.edu, Tel: 404-880-6433

2 – Network Computing Services, Inc., Army HPC Research Center
1200 Washington Ave. S., Minneapolis, MN 55415
Email: ajohn@networkcs.com, Tel: 612-337-3415

3 – Engineer Research and Development Center
Coastal and Hydraulics Laboratory, ERDC-CHL
3909 Halls Ferry Road, Vicksburg, MS 39180-6199
Email: berger@HL.wes.army.mil, Tel: 601-634-2823

ABSTRACT

In this paper, we present some recent results in numerical simulation of waves interacting with ships in motion. In our approach, the governing equations are the Navier-Stokes equations written for multiple incompressible fluids. We solve these equations over a non-moving mesh. An interface function with two distinct values serves as a marker identifying the location of the free-surface. This function is transported throughout the computational domain with a time-dependent advection equation. The stabilized finite element method is used to discretize the governing equations. The finite element formulations are implemented in parallel using MPI. High performance computing tools and optimized techniques are utilized to solve applications on unstructured meshes with 200 million tetrahedral elements. Our web-based client-server technology is used to visualize the very large data sets.

1. INTRODUCTION

Ship hydrodynamics is one of the fundamental fluid dynamics problems. In these problems, the free-surface of the water interacts with the hull, and as a result, waves are formed in the front, side and wake of the ship. Depending on the ship velocity and water depth, the strength and behavior of the waves can change. For example, in shallow waters, even at relatively low speeds, strong waves are formed. The main parameter distinguishing wave behavior is the Froude number defined as:

$$F_r = \sqrt{\frac{V_\infty^2}{Hg}} . \tag{1}$$

Here V_∞ is the ship velocity, H is the length of the ship, and g is the gravitational acceleration. The flow is characterized as subcritical, critical and supercritical if the Froude number is less than one, equal to one, and greater than one, respectively. In ship hydrodynamics, we rarely expect critical and supercritical conditions. In most cases, the Froude number is less than 0.5 (subcritical condition) [1].

An effective ship design requires accurate knowledge of the force distribution along the hull. The forces that the water generates on the hull will tell the designers how strong the hull needs to be at each location, the drag, the power required to propel the ship and the integrity of the hull. Waves are one of the main factors contributing to force distribution. Therefore, accurate estimates of their shape, location, and size are critical. Traditionally, scaled models of the ship's hull are created and then experiments are performed using these models in water tanks. These types of studies are very expensive and time consuming.

Computer modeling is a useful tool for studying flow behavior around ships. Numerical simulations can help reduce the cost and the margin of error associated with traditional studies. Also, many different designs can be tested in very short periods of time. Numerical simulations of free–surface flow applications are based on the solution of a complex set of partial differential equations governing the conservation of mass and momentum. These equations are referred to as the Navier-Stokes equations. In our approach, we use stabilized finite element methods to solve the governing equations. The Navier-Stokes equations are written over a discrete domain made of millions of small elements. These equations are integrated over each element, resulting in a set of coupled nonlinear equations, which are solved iteratively to obtain the velocity, pressure, and the location of the free-surface over the discrete domain.

Generally, there are two distinct approaches in numerical simulation of free-surface flows. Depending on the physical characteristics of the problem, either moving mesh or fixed mesh methods are used [2-3]. In the moving mesh method, the nodal coordinates on the free-surface are moved to track the motion of the free-surface. Here, the computational mesh needs to change to account for the motion of the free-surface as well. This method can cause large element distortions in the mesh when the geometry is too complex [4-5]. As the element distortion grows to an unacceptable limit, generation of a new mesh and projecting the solution from the old mesh to the new one is required. In complex 3D free-surface flow applications, this procedure is extremely difficult and time consuming. In such cases, computations over fixed meshes is more feasible.

The free-surface flow computations over fixed meshes is based on a volume-of-fluid (VOF) approach [6]. In the fixed mesh methods, the Navier-Stokes equations are solved over a non-moving mesh, and a free-surface function with two distinct values is used to identify the actual location of the free-surface. This free-surface function is transported throughout the computational domain with a time-dependent advection equation. In this method, the mesh resolution becomes a prime factor in determining accuracy because this mesh resolution determines how accurately the free-surface function is represented [1-3, 7-10]. Of course, by using this method we eliminate the need and computational overhead of actually moving the mesh to accommodate the mesh motion, and this is a great advantage.

In fixed mesh methods, mass conservation is one of the main concerns. Numerical errors usually result in non-physical gains and losses in mass (i.e. liquid volume) during the computations. For long-term simulations, large inaccuracies can build-up. To eliminate this problem, we have designed and implemented the IS-GMC (Interface-Sharpening/Global Mass Conservation) algorithm [2]. This algorithm is based on accurately advecting the free-surface function in such a way that fluid mass is conserved. We have verified the accuracy of this method with many test problems [2, 8-9]. We use our fixed mesh method with built-in IS-GMC to simulate application in this article.

Numerical simulations of free-surface flows are large scale. We have developed various high performance computing (HPC) tools to handle large-scale free-surface flow applications. This includes a geometry modeler and an automatic mesh generator [4-5], highly scaleable and optimized parallel finite element flow solver [8-9,11], and web-based client-server technology for remote visualization [8]. These HPC tools have provided us a unique capability to simulate applications with meshes consisting of 200 million elements. These HPC tools are highly optimized for both memory and parallel scalability especially for such large applications.

In this paper, we present our recent results in numerical simulation of waves interacting with ship in motion. We provide sufficient details of techniques we use to solve and visualize very large-scale applications on unstructured meshes with up to 200 million tetrahedral elements.

2. GOVERNING EQUATIONS

Let us denote Ω and $(0,T)$ as the space and time domains, respectively. The boundary of Ω is defined by Γ. Using this notation, the governing equations are written as:

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \mathbf{g}) - \nabla\cdot\boldsymbol{\sigma} = 0 \qquad \text{on } \Omega \quad \forall t \in (0,T), \tag{2}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{on } \Omega \quad \forall t \in (0,T), \tag{3}$$

where

$$\boldsymbol{\sigma} = -\mathrm{p}\,\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(u), \tag{4}$$

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}). \tag{5}$$

Here **u**, p, ρ, **g**, and μ are the velocity, pressure, density, gravitational force, and dynamic viscosity, respectively. The strain tensor is denoted by $\boldsymbol{\varepsilon}$ and **I** represents the identity tensor. Equations (2-3) are completed by an appropriate set of boundary and initial conditions.

To simulate free-surface flow problems, we assume there are two fluids interacting with each other where the effect of one fluid on the other one is very small. For example, in water flows, air is constantly interacting with water. Since the density of air is almost 1000 times less than water, its effect on the water is negligible.

The interface function ϕ has two distinct values (0,1) and is used to differentiate between the two fluids. A time-dependent advection equation transports this function throughout the computational domain with the fluid velocity

$$\frac{\partial \phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0 \qquad \text{on } \Omega \quad \forall t \in (0,T). \tag{6}$$

Using ϕ, the density and viscosity can be calculated as:

$$\rho = \phi\rho_A + (1-\phi)\rho_B\,, \tag{7}$$

$$\mu = \phi\mu_A + (1-\phi)\mu_B\,, \tag{8}$$

where the subscripts A and B denote the fluid A and fluid B. Initially, ϕ is set to 0 in Fluid A and 1 in Fluid B.

3. FINITE ELEMENT FORMULATIONS

The finite element formulations are based on the SUPG (stabilized-upwind/Petrov-Galerkin) and PSPG (pressure-stabilized/Petrov-Galerkin) techniques. The SUPG term allows us to solve flow problems at high speeds [11-13] and the PSPG term eliminates instabilities associated with the use of linear interpolation functions for both pressure and velocity [2,9].

In the finite element formulation, we first define appropriate sets of trail solution spaces $S^h_{\mathbf{u}}$, S^h_p, and S^h_ϕ, and weighing function spaces $V^h_{\mathbf{u}}$, V^h_p and V^h_ϕ for the velocity, pressure, and the free-surface function, respectively. The stabilized finite element formulation of Equations (2,3,6) can then be written as follows: for all $\mathbf{w}^h \in V^h_{\mathbf{u}}$, $q^h \in V^h_p$, and $\psi^h \in V^h_\phi$, find $\mathbf{u}^h \in S^h_{\mathbf{u}}$, $p^h \in S^h_p$, and $\phi^h \in S^h_\phi$ such that:

$$\begin{aligned}
&\int_\Omega \mathbf{w}^h \cdot \rho[\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\nabla\mathbf{u}^h - \mathbf{g}]d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h,\mathbf{u}^h)d\Omega \\
&+ \int_\Omega q_p^h \nabla\cdot\mathbf{u}^h d\Omega + \int_\Omega \psi^h(\frac{\partial\phi^h}{\partial t} + \mathbf{u}^h\cdot\nabla\phi^h)d\Omega + \sum_{e=1}^{ne}\int_{\Omega^e}\tau_c\nabla\cdot\mathbf{w}^h\rho\nabla\cdot\mathbf{u}^h d\Omega \\
&+ \sum_{e=1}^{ne}\int_{\Omega^e}\frac{\tau_m}{\rho}\left[\mathbf{u}^h\cdot\nabla\mathbf{w}^h - \nabla\cdot\boldsymbol{\sigma}(q_p^h,\mathbf{w}^h)\right]\cdot\left[\rho[\frac{\partial\mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\nabla\mathbf{u}^h - \mathbf{g}] - \nabla\cdot\boldsymbol{\sigma}(p^h,\mathbf{u}^h)\right]d\Omega \\
&+ \sum_{e=1}^{ne}\int_{\Omega^e}\tau_i\nabla\psi^h\cdot\nabla\phi^h d\Omega = \int_{\Gamma_u^h}\mathbf{w}^h.\mathbf{h}\,d\Gamma\,.
\end{aligned} \tag{9}$$

The parameters τ_c, τ_m, and τ_i are the stabilization parameters defined as:

$$\tau_m = \left[\left(\frac{2\,\|\mathbf{u}\|}{h}\right)^2 + \left(\frac{4\mu}{\rho h^2}\right)^2\right]^{-\frac{1}{2}}, \tag{10}$$

$$\tau_c = \frac{h}{2}\|\mathbf{u}\|z, \qquad z = \begin{cases} R_u/3 & R_u \le 3 \\ 1 & 3 < Ru \end{cases}, \tag{11}$$

$$\tau_i = \frac{h}{2}\|\mathbf{u}\|, \tag{12}$$

where h is the element length and R_u is the cell Reynolds number [2,11].

In Equation (9), the first four integrals together with the right-hand-side represent the Galerkin formulation of the governing equations. The first, second and third series of element-level integrals in the formulation are the least-square stabilization of the continuity equation, the SUPG and PSPG stabilization for momentum equations, and the artificial diffusion stabilization for the interface function, respectively. The diffusion formulation for the interface function eliminates the numerical undershoots (below 0) and overshoots (above 1) of the interface function around the interface. In the IS-GMC algorithm, we recover the sharpness of the interface in such a way that the global mass conservation for each fluid is enforced.

4. INTERFACE-SHARPENING/GLOBAL MASS CONSERVATION (IS-GMC)

The finite element formulation in Equation (9) introduces numerical diffusion for the interface function ϕ. In IS-GMC, to recover the sharpness of the interface function, after each iteration, ϕ is replaced by ϕ_{new} as following:

$$\phi_{new} = \begin{cases} \beta^{(\alpha-1)}\phi^{\alpha} & 0 \le \phi \le \beta \\ 1 - (1\text{-}\beta)^{(\alpha-1)}(1-\phi)^{\alpha} & \beta \le \phi \le 1 \end{cases}, \tag{13}$$

where 1.2≤α≤1.5 is a sharpening parameter [9], and 0≤β≤1 is a point satisfying the global conservation of mass for each fluid.

To determine β, we satisfy the mass conservation at a given time t for ϕ_{new}. Therefore:

$$\rho_A \int_{\Omega} \phi_{new} d\Omega = m_A + \rho_A \int_t \int_{\Gamma} \phi\, \mathbf{u.n}\, d\Gamma dt\,, \tag{14}$$

$$\rho_B \int_{\Omega} \phi_{new} d\Omega = m_B + \rho_B \int_t \int_{\Gamma} (1-\phi)\, \mathbf{u.n}\, d\Gamma dt\,, \tag{15}$$

where m_A and m_B are the initial mass of Fluid A and Fluid B, respectively. Note that we only need to satisfy either Equation (14) or (15). Combining Equations (13) and (14) and assuming that the parameter α is given and constant, we obtain

$$M\beta^{(1\text{-}\alpha)} + N(1\text{-}\beta)^{(1\text{-}\alpha)} = R\,, \tag{16}$$

where M, N, and R are all functions of β. This nonlinear equation is solved using a Newton-Raphson algorithm. Typically, with the initial guess of 0.5, the algorithm converges in three iterations [2].

5. ITERATIVE SOLUTION STRATEGY

The discretization of the finite element formulation results in a series of coupled, nonlinear systems of equations that need to be solved at every time step. First, we linearize the finite element formulation, Equation (9), using Newton-Raphson method. Then, the resulting series of linear systems of equations are solved also iteratively using the GMRES algorithm [14]. For very large systems of equations, we use a matrix-free iteration strategy to obtain the solution of the nonlinear system. This element-vector-based computation totally eliminated the need to form any matrices, even at the element level [11].

6. PARALLEL IMPLEMENTATION

The computations of 3D free-surface flow applications are large scale. Parallel supercomputers with hundreds of fast processors, such as CRAY T3E and IBM SP are used to reduce the computing time [1-5, 7-11]. In the parallel implementation, we use a message-passing computing paradigm, making the cross-processor communication explicit. This is accomplished by using the MPI (Message Passing Interface) libraries. Prior to the computation, the finite element mesh is partitioned into contiguous subdomains [15], and these subdomains are assigned to individual processors. To ensure load balancing for each processor, each subdomain contains approximately the same number of elements. Element-level computations are carried out independently for each processor [11]. Data transfer between the elements and nodes is accomplished in two steps. First, data is gathered or scattered between the elements and the nodes residing on the same processor. This step does not involve any communication. At the second step, the gather and scatter operations are performed to exchange data across the processors only for those nodes residing on the boundary of subdomains. The two-step gather and scatter operations lead toward more efficient communication strategy.

7. AUTOMATIC MESH GENERATION

Our automatic mesh generation tools have been under development at the Army HPC Research Center (AHPCRC) for quite some time, and have been used in a wide variety of applications. The automatic mesh generator is not a single application, but consists of a series of three applications including a 3D geometric modeler, an automatic surface mesh generator, and an automatic 3D volume mesh generator. For details, see [4,5].

8. PARALLEL MESH MULTLIPICATION

In one specific application, we use an unstructured finite element mesh with almost 200 million tetrahedral elements. Generating such a large mesh with our automatic mesh generator created unique challenges due to the fact that the largest mesh that could be created on a workstation is roughly 3 million elements. Due to this constraint, we had to develop a parallel mesh multiplication program in order to turn a 3 million element mesh into a 200 million element mesh. We believe that such large meshes, which are becoming more common for modern HPC resources, should be created with such a mesh multiplication technique. An automatic mesh generator is capable of handling complex geometry and making high quality meshes, but not necessarily useful at just adding more and more nodes in order to increase the number of elements to reach a desired size. A two-step technique, as was used in some of the simulations here, is a better approach to generating extremely large unstructured meshes [8].

The mesh multiplication technique is based on dividing each tetrahedral element into 8 new sub-elements. The entire mesh size is increased by a factor of 8 in total element size. The various steps of the program are listed below:

- Generate the individual edges of the 3D tetrahedral element mesh.
- Divide each edge into two new edges through the creation of a new nodal point at the edge's mid-point.
- Divide each element into 8 new elements using the new nodal points that were created at the center of each edge (see Figure 1).
- Re-compute the dual (i.e. the element to element connections) and boundary information based on this new element arrangement.

Of course, this algorithm is rather simple to implement on a serial (i.e. single processor) computer, but the size of the new (i.e. divided) mesh becomes very large rather quickly, and will usually require more memory than a single processor workstation has available. To overcome the memory limitation, we implemented this algorithm in parallel using MPI so that we can have access to the large memory usually available on such HPC systems.

The main difficulty in a parallel implementation of this algorithm is the fact the mesh is "physically" distributed amongst the processors which creates complications in the creation and storage of the unique edges of the mesh in Step 1. Complex inter-processor communication is required. Also, the reconfiguring of the dual information in Step 4 requires more inter-processor interaction. Both of these difficulties were overcome, and an effective parallel mesh multiplication program was created and used in the simulations presented here.

In our "preliminary" program, we neglected the possible curved shape of the boundaries of the mesh. The location of the new nodal points on edges located on curved boundaries was simply created at the mid-point, thus ignoring the possible curvature of that boundary. We assumed that the initial meshes used were of sufficient refinement to accurately represent any curvature of the boundaries, so the effect of a simple mid-point division could be neglected. In future versions, we plan to take the curvature of the boundaries into consideration in creating new nodes on model boundaries.

9. LARGE SCALE DATA VISUALIZATION

The visualization of such large data sets such as the ones used here posed unique challenges due to their raw size. A typical workstation does not have nearly the memory or computing power needed to process the data set alone. Furthermore, the transport and storage of this large data set (up to 100 Gigabytes in our applications) from the parallel computing server to the user's desktop system can become a very time-consuming and possibly impossible task. To overcome this bottleneck, we used the Presto Visualizer, developed at the Army HPC Research Center to perform the data visualization used in the examples presented here.

This data visualizer is written in a client-server framework [7-9]. The server part of the program runs in parallel (MPI-based) on the computer where the data is located, and in this case, 1024 processors of the CRAY T3E were used. This server program is responsible for loading the large 3D data set, processing it, and extracting any visualization constructs requested by the client program. The client program runs on a user's desktop system and is responsible for all user interaction and displaying all 3D geometry sent to it by the server. The two programs communicate with standard Internet protocols. The program can visualize the 3D data set through surface (i.e. boundary) shadings, cross-sections, iso-surfaces, and streamlines, and that is the key to the effectiveness of this client-server model. The visualization constructs (i.e. the geometry that actually gets shown on the computer screen) are all based on "2D" or flat surface geometry, and the size of these surfaces is an order of magnitude less than the raw 3D simulation data that the parallel server program is processing. This surface data can more easily be sent across a moderately configured Internet connection and be displayed on the client workstation with the OpenGL 3D graphics library.

10. NUMERICAL EXAMPLES

We apply these techniques to simulate breaking waves, flow in contraction channels at super-critical condition, and waves interacting with ships in motion.

Breaking Waves. Waves approaching the coast increase in steepness as the water depth decreases. When the wave height is approximately equal to the water depth, the wave breaks, dissipating wave energy and inducing longshore currents, increasing in the mean water level, and suspending sediment. The surf zone is the region from the shoreline to the seaward boundary of wave breaking. Within the surf zone, wave breaking is the dominant feature. The surf zone is the most dynamic coastal region with sediment transport and bathymetry change driven by breaking waves and wave-induced currents. Surf zone wave conditions are required for estimating potential storm damage (flooding and wave damage), calculating shoreline evolution and cross-shore beach profile change, designing coastal structures and beach fills, and developing shoreline management policy.

Wave breaking has generally been parameterized using similarity methods [16], energy dissipation methods in the form of a hydraulic jump [17] or relaxation towards a stable wave height [18]. Wave irregularity is included by assuming wave heights are Rayleigh distributed [19]. These simple methods provide reasonable representation of wave heights in the surf zone, but do not directly provide estimates of other wave characteristics in the surf zone (e.g., distribution of turbulence, evolution of spectral shape, nonlinearities, effects of currents or multiple wave trains). Accurate computational tools are required to model the detailed surf zone hydrodynamics and sediment transport processes.

Here we simulate wave formation and wave breaking in a 2D domain as shown in Figure 2. Initially, the maximum water height on the left side (H) is 1.0 m. The x-coordinate of computational domain varies from 0 to 25 meters. The y-coordinate on the left side is between 0 and 1.75 meters. The slope of the bottom is 1/20.

To create waves, we constantly accelerate and decelerate the computational domain with the prescribed sinusoidal function:

$$a_x = -a_{\max} \sin\left(\frac{2\pi t}{T}\right), \tag{17}$$

where $a_{\max}$ is the maximum acceleration, T is the period and t is the elapsed time. In this simulation, $a_{\max}$ and T are 0.49 m/s^2 and 5s, respectively. Integrating this equation twice, we obtain the motion of the computational domain:

$$s = a_{\max} \frac{T^2}{4\pi^2} \sin\left(\frac{2\pi t}{T}\right). \tag{18}$$

Therefore, the x-coordinate of noninertial frame is simply obtained by adding s to X. Although the problem is 2D, the simulations are carried out using a 3D multifluid flow solver. The 3D finite element mesh is made by using 1,500, 70 and 1 elements in x, y, and z directions, respectively. This results in a finite element mesh with 213,142 nodes and 105,000 hexahedral elements. Simulations start with still water at $t = 0$. The time step is set to 0.0083333s. At every time step, a coupled nonlinear system of equations with more than 840,000 unknowns is solved iteratively using matrix-free GMRES technique. The number of inner iterations is 50. The simulations are carried out on CRAY T3E with 16 processors. The series of the frames in Figure 3 show the history of breaking waves with water and air shown in red and blue colors, respectively.

The way we generate waves in not natural and this problem is a set up for demonstration purpose only. A more rigorous approach will be used in actual studies.

Contraction Channel. The contraction channel walls are composed of two equal circular arcs each having a radius of 75 inches (see Figure 4). Water at high velocity enters the 24-inch wide channel at the speed of 85.3 inch/s and passes the narrow section of the channel, which is 12 inches wide. The Froude number with respect to the entering water elevation of 1 3/16 inch is 4.0. Here, we use a mesh consisting of 2,715,171 nodes and 2,640,000 hexahedral elements. The number of elements in the axial, vertical and cross-flow directions are 401, 61 and 111, respectively. The channel lengths before and after the contraction section are 19 11/16 inch and 136 inch, respectively.

The computation starts with an initial uniform velocity field equal to the entering water velocity. The time step is set to 0.00346 s. As computations continue, water waves are formed in the contraction section and reflected back into the narrow section of the channel from the channel walls. The two pictures in Figure 5 show the comparison of the computed water elevation along the channel wall and center to experimental data. All the flow features observed from the experiments are also captured in the computation. Here the computed results are very comparable with the experimental data. Figure 6 shows the water surface in the channel at t = 2.56 s, which is sufficient time to reach steady-state surface waveforms.

Hydrodynamics of Two Ships. In this problem, we apply our numerical techniques to simulate two ships traveling in a narrow channel. From this simulation, we can obtain forces produced by the interaction of passing ships. This in turn can be used to represent the ship maneuverability and so improve navigation channel design. Also, the size and location of the waves created by each ship, as well as the interaction between the two wakes, are important to study.

Here we simulate two "idealized" ship models traveling side-by-side down a narrow channel. The two ship models are identical, but one travels slightly behind the other (see Figure 7). Both ships have identical speed. The idea here is to try to determine the effect the wake and drawdown of the first ship will have on the trailing ship. With our mesh multiplication technique, we generate three levels of mesh refinement. The first mesh, called the Level-1 mesh, was created directly by the automatic mesh generator and contained 3.12 million elements and 515 thousand nodal points. Figure 7 also shows the surface grid of one of the ships in the Level-1 mesh. We then applied our mesh multiplication technique to derive the Level-2 mesh which contained 25 million elements and 4.2 million nodes. We then applied mesh multiplication one final time to create the Level-3 mesh, which contains 200 million elements and 33.6 million nodes. We ran numerical simulations on all three meshes under the exact same flow conditions, and provide pictures here of results of the Level-2 and Level-3 runs. It took between 150 and 200 time steps for the wake to develop, and on 1024 processors of the CRAY T3E-1200, a single time step could be calculated in 0.67 minutes for the Level-2 mesh, and 3.02 minutes for the Level-3 mesh.

Figure 8 shows the pressure distribution on the surface of the two ships for the Level-3 (200 million element) run. The water height (shown in red) on the surface of the two ships for the Level-3 run is also shown in Figure 8. An iso-surface showing the water height for the Level-3 run could not be created with our data visualizer because the memory on the client computer, which has one Gigabytes of memory, was not sufficient to process this triangular surface topology. We do show an iso-surface designating the water height for the Level-2 mesh in Figure 9. A view of the ships and the water height from underneath is also shown in Figure 9.

The results were very interesting, and the waves that were created are quite detailed. The rather large bow waves created from the front of the ships can clearly be seen, as well as other smaller waves created on the side and in the wake of the two ships. It can be seen that there is some interaction between the wakes of the two ships.

DTMB 5415. Here we apply our numerical techniques to simulate water flow around the US Navy Combatant, DTMB 5415. Model 5415 was conceived as a preliminary design for a Navy surface combatant ca. 1980 [20]. This model was also selected in the Gothenburg 2000 workshop [20]. Figure 10 shows the geometric model of the DTMB 5415. For the scaled model, the experimental data already exists for two speeds, 4 and 6 knots. The hull geometry includes both a sonar dome and a transom stern. The length of the ship is 18 7/10 feet long. To validate our flow solver, the boundary conditions are exactly the same as the ones used in experiments.

The Froude numbers, based on the speeds of 4 and 6 knots are 0.28 and 0.41, respectively. We carried out simulations for both Froude numbers on the Cary T3E-1200 with 256 processors. Two unstructured meshes were used, one with more than one billion tetrahedral elements and the other with 25 million tetrahedral elements.

However, the results reported in this article corresponds for the mesh with 25 million elements. The pictures in Figure 11 show the coarsened mesh and the element processor distribution on the surface of the DTMB 5415.

Figure 12 shows the water elevation (red color) along the hull for Froude number 0.28. The mesh is detailed enough to capture the vortex shedding generated from the sonar dome. Figure 13 shows this vortex shedding with x-component of the velocity along the symmetry plan. To measure the accuracy of the computations, in Figure 14, we compare the nondimensional water elevation along the hull with available experimental data. The computed results are in very good agreement with experimental data.

We also carried out the computations for the Froude number of 0.41 using the same finite element mesh. In this case, the water elevation is the same as the one in Froude number 0.28, but ship is moving faster. Figures 15 shows the water elevation along the hull for Froude number 0.41. In this case, stronger waves are formed in the front of the ship, causing the water to rise higher, compared to Froude number 0.28. Also, as we anticipated, stronger vorticities are generated from the sonar dome, as can be seen in Figure 16. In Figure 17, we also compare the nondimensional water elevation along the hull with available experimental data for Froude number 0.41.

11. CONCLUDING REMARKS

We developed high performance computing tools and techniques to simulate waves interacting with ships in motion. We outlined the details of the techniques we use to solve applications on unstructured meshes with more than one billion tetrahedral elements. We used client-server based Presto Visualizer, developed at the AHPCRC to interactively visualize the very large data sets generated by these simulations. We applied these techniques to simulate breaking waves, flow in contraction channels at super-critical condition, and waves interacting with ships in motion. We demonstrated the accuracy of the computations by comparing the results with experimental data. The computed results are in most cases in very good agreement with experimental data.

Acknowledgment

Work funded by the DoD High Performance Computing Modernization Program ERDC Major Shared Resource Center through Programming Environment and Training (PET) contract number DAHC 94-96-C0002. Partial support is also provided by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory contract number DAAH04-95-C-0008. Views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense position, policy, or decision unless so designated by other official documentation.

References

1. S. Aliabadi and S. Shujaee, "**Two-Fluid Flow Simulations Using Parallel Finite Element Method**", accepted for publication in the *Journal of the Society for Computer Simulation International*.

2. S. Aliabadi and T. Tezduyar, "**Stabilized-Finite-Element/Interface-Capturing Technique for Parallel Computation of Unsteady Flows with Interfaces**", *Computer Methods in Applied Mechanics and Engineering*, 190, 243-261 (2000).

3. T. Tezduyar and S. Aliabadi, "**EDICT for 3D Computation of Two-Fluid Interfaces**", *Computer Methods in Applied Mechanics and Engineering,* **190** (2000) 403-410.

4. A.A. Johnson and T.E. Tezduyar, "**Parallel Computation of Incompressible Flows with Complex Geometry**", *International Journal for Numerical Methods in Fluids*, **24**, 1321-1340 (1997).

5. A. Johnson and T. Tezduyar, "**Advanced Mesh Generation and Update Methods for 3D Flow Simulations**", *Computational Mechanics*, **23**,130-143 (1999).

6. W. Hirt and B. D. Nichols, "**Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries**", *Journal of Computational Physics*, 39:201-225 (1981).

7. S. Aliabadi, A. Johnson, B. Zellars, A. Abatan and C. Berger, "**Remote Simulation and Visualization of Free-Surface Flow Problems**", *Proceeding of the 2000 Users Group Conference, DOD High Performance Computing Modernization Program,* Albuquerque, NM, June 2000.

8. A. Johnson and S. Aliabadi, "**Application of Automatic Mesh Generation and Mesh Multiplication Techniques to Very Large Scale Free-Surface Flow Simulations**", *Proceeding of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations,* Whistler, British Columbia, Canada, September 2000.

9. S. Aliabadi and A. Johnson, "**Large-Scale Parallel Simulation of Free-Surface Flow Applications**", *Proceeding of the 21st Latin American Congress on Computational Methods in Engineering, Cilamce 2000, Institute of Pure and Applied Mathematics,* Rio de Janeiro, Brazil, December 2000.

10. T. Tezduyar, S. Aliabadi and M. Behr, "**Enhanced-Discretization Interface-Capturing Technique (EDICT) for Computation of Unsteady Flows with Interfaces**", *Computer Methods in Applied Mechanics and Engineering*, **155**, 235-248 (1998).

11. S. Aliabadi and T. Tezduyar, "**Parallel Fluid Dynamics Computations in Aerospace Applications**", *International Journal for the Numerical Methods in Fluids*, 21:783-805 (1995).

12. S. K. Aliabadi and T. E. Tezduyar, "**Space-time Finite Element Computation of Compressible Flows Involving Moving Boundaries and Interfaces**", *Computer Methods in Applied Mechanics and Engineering*, **107**, 209-223 (1993).

13. S. K. Aliabadi, S. E. Ray and T. E. Tezduyar, "**SUPG Finite Element Computation of Viscous Compressible Flows Based on the Conservation and Entropy Variables Formulations**", *Computational Mechanics*, **11**, 300-312 (1993).

14. Y. Saad and M. Schultz**,** "**GMRES: Generalized Minimal Residual Algorithm for Solving Nonsymmetic linear Systems**", *SIAM Journal of Scientific and Statistical Computing*, **7**, 856-896, 1986.

15. G. Karypis and V. Kumar, "**Parallel Multilevel k-Way Partitioning Scheme for Irregular Graphs**", SIAM Review, **41 (2)** , 278-300 (1999).

16. M. S. Longuet-Higgins and R. W. Stewart, "**A Note on Wave Setup**", *J Marine Research* 21(1), 4-10 (1963).

17. B. LeMehaute, "**On the Nonstaturated Breaker Theory and the Wave Run Up**", *8th Coastal Engineering Conference*, ASCE, 77-92 (1962).

18. W. R. Dally, R. G. Dean and R. A. Dalrymple, "**Wave Height Variation across Beaches of Arbitrary Profile**", *J Geophysical Research* 90(C6), 11917-11927 (1985).

19. W. R. Dally, "**Random Breaking Waves: a Closed-Form Solution for Planar Beaches**", *Coastal Engineering* 14(3), 233-263, 1990.

20. http://www.iihr.uiowa.edu/gothenburg2000/5415/combatant.html

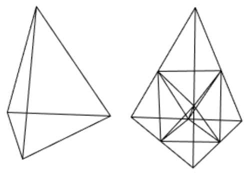


Figure 1. A single tetrahedral element divided into eight sub-elements.

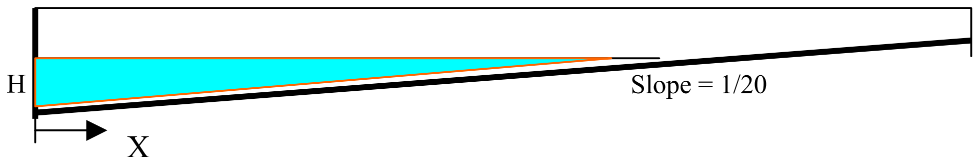


Figure 2. Wave formation and wave breaking in a 2D domain.

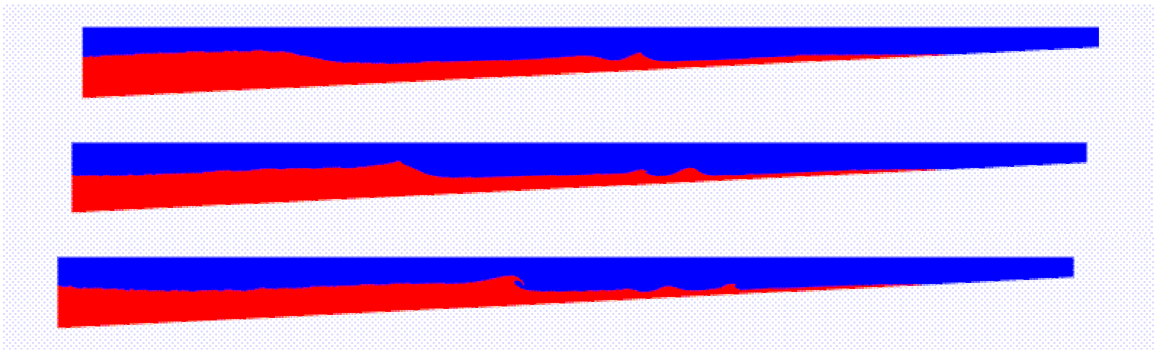


Figure 3. The series of the frames show the history of breaking waves with water and air shown in red and blue colors, respectively.

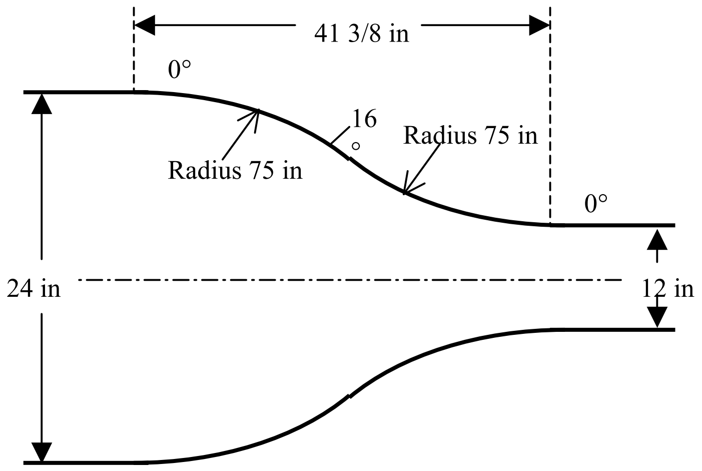
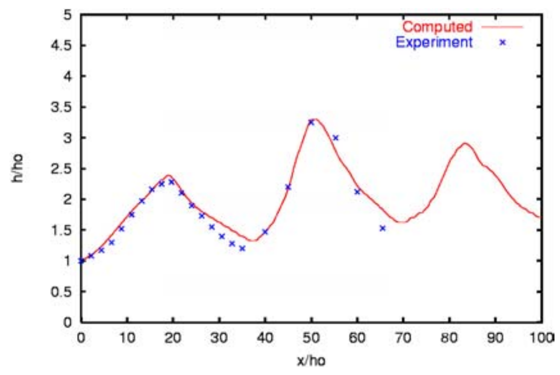


Figure 4. The contraction channel.



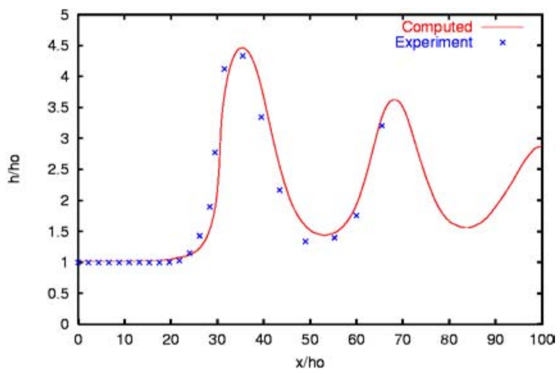
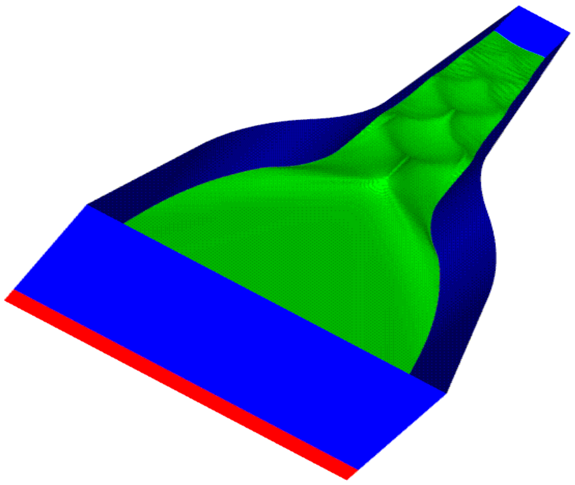


Figure 5. Comparison of the computed water elevation along the channel wall (left picture) and the channel center (right picture) to experimental data.



Figures 6. The water surface in the channel at t = 2.56 s.

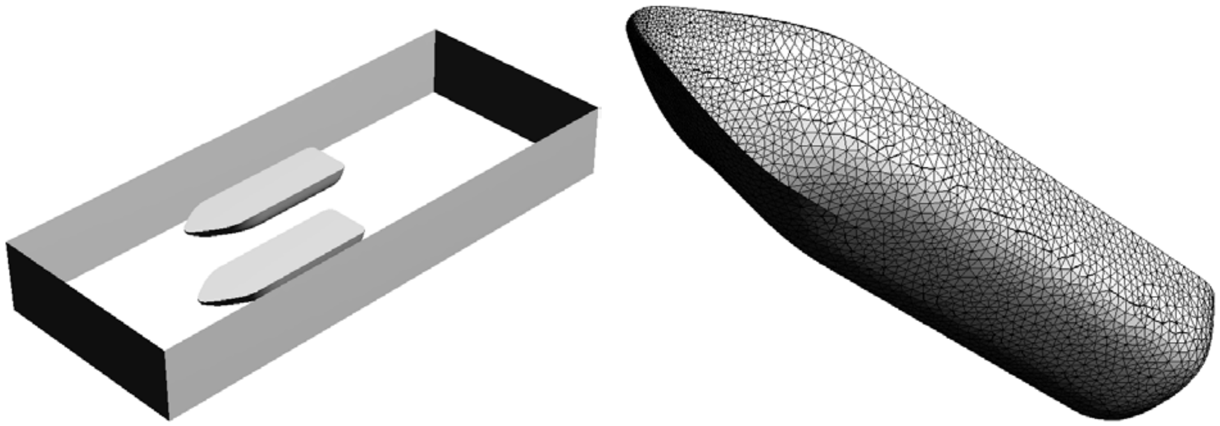


Figure 7. The computational domain (left), and the surface mesh of one of the ships (right).

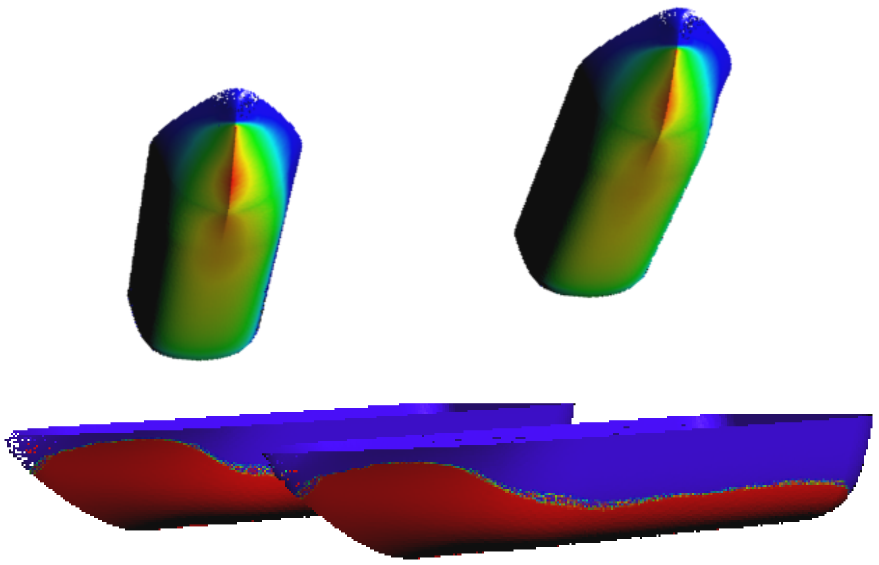


Figure 8. The pressure field on the surface of the two ships for the Level-3 mesh (top), and the free-surface profile on the sides of the ships (bottom). The visualization artifacts located at the tip of the ships is caused by the graphics engine's failure to display such small triangles.

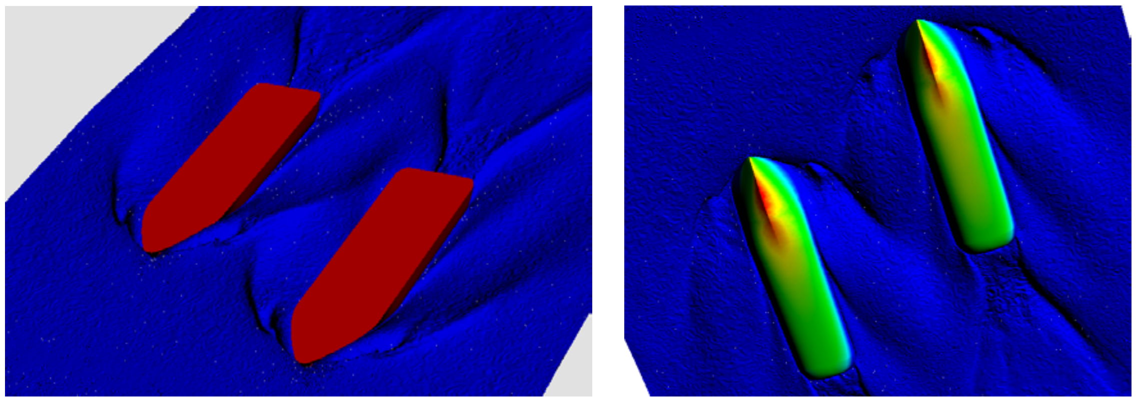


Figure 9. The surface of the water as viewed from above (left) and below (right). This data was generated for the Level-2 mesh.

Figure 10. The geometry model of the DTMB 5415.

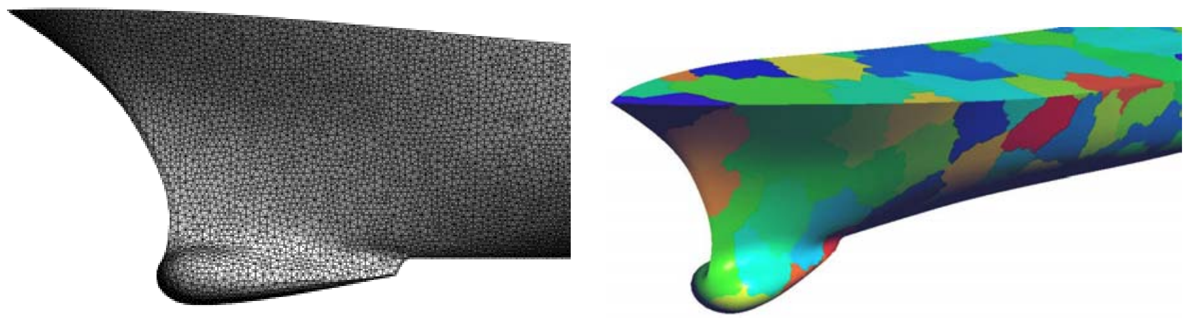


Figure 11. The coarsened mesh and partitioned mesh on the surface of the DTMB 5415.

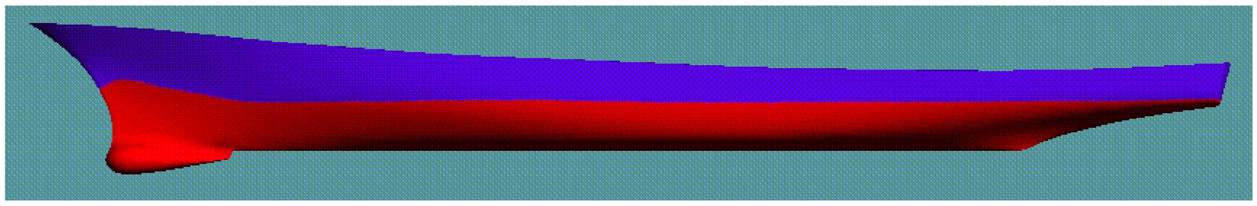


Figure 12. The water elevation (red color) along the hull for Froude number 0.28.

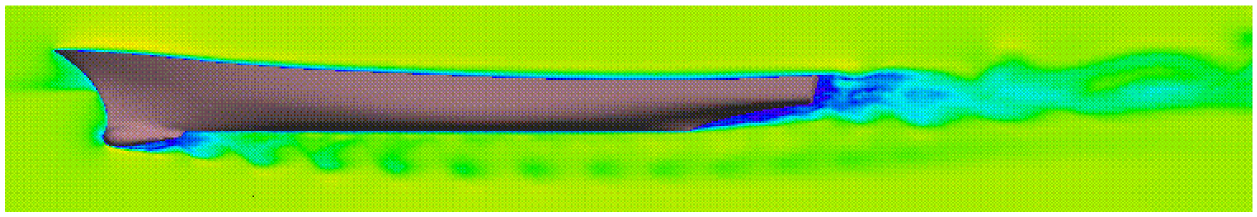


Figure 13. The vortex along the symmetry plan for Froude number 0.28. The colors show the x-component of the velocity.

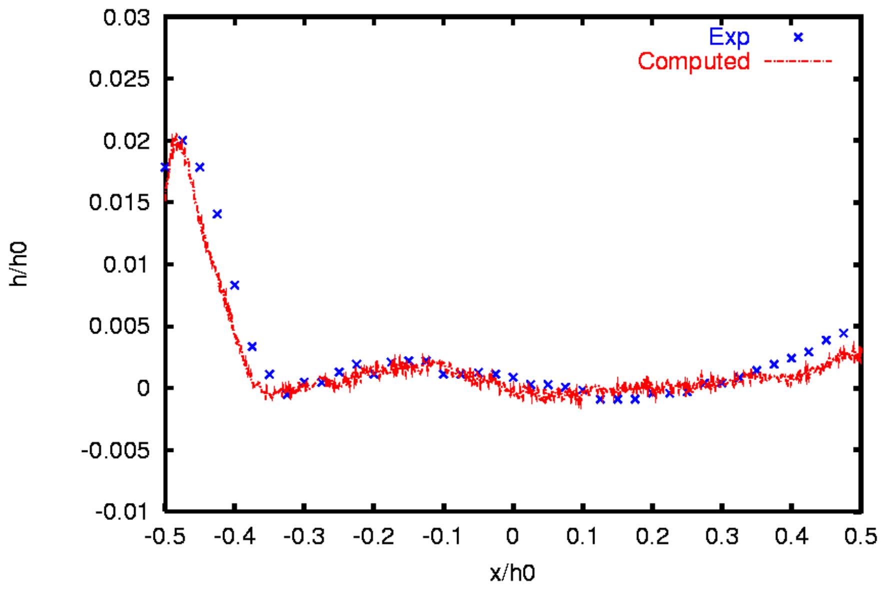


Figure 14. Comparison between the computed and measured water elevation along the hall for Froude number 0.28.

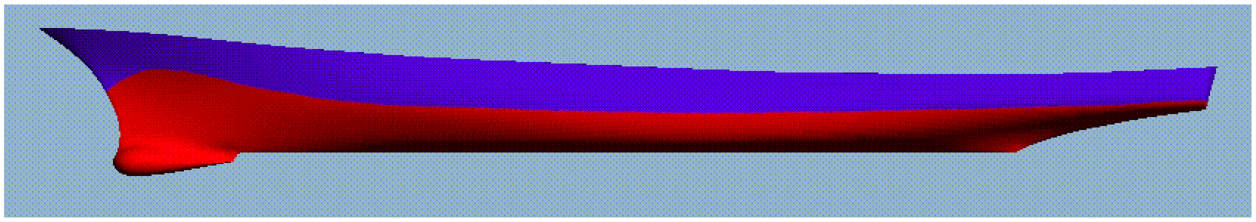


Figure 15. The water elevation (red color) along the hull for Froude number 0.41.

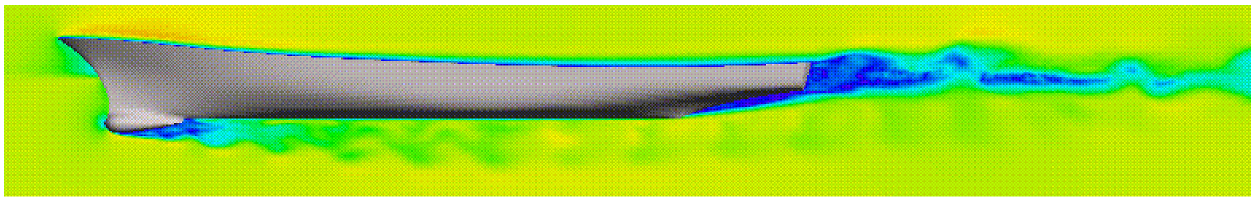


Figure 16. The vortex along the symmetry plan for Froude number 0.41. The colors show the x-component of the velocity.

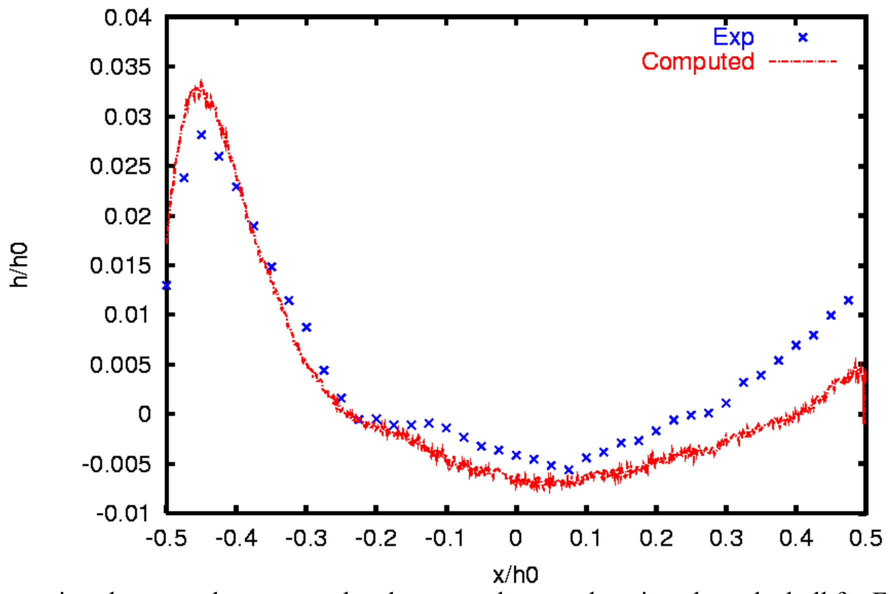


Figure 17. Comparison between the computed and measured water elevation along the hull for Froude number 0.41.